Inside SEER-SEM

Lee Fischman, Karen McRitchie, and Daniel D. Galorath
Galorath, Inc.

The System Evaluation and Estimation of Resources - Software Estimating Model (SEER-SEM) is a commercially available software project estimation model used within defense, government, and commercial enterprises. Introduced over a decade ago and now in its seventh release, it offers a case study in the history and future of such models. SEER-SEM and its brethren are built upon a mix of mathematics and statistics; this article provides insight into its inner workings and basis of estimation.

If you follow the roots of software estimation models, you will find many have common ancestors. The System Evaluation and Estimation of Resources - Software Estimating Model (SEER-SEM) began with the Jensen model and diverged significantly in the early 1990s. Barry Boehm's Constructive Cost Model work provided for the redefinition of some of the original Jensen model parameters into SEER-SEM. Don Reifer and Dan Galorath's work on the NASA Softcost model also found its way into SEER-SEM in addition to Halstead's software science metrics. The Jensen model itself was first calibrated using some of the same data as the Putnam model. Earlier work by Doty Associates introduced the idea of factoring in development environment influences via parameters. Work on this model continues today.

SEER-SEM's Architecture

SEER-SEM is composed of a group of models working together to provide estimates of effort, duration, staffing, and defects. These models can be briefly described by the questions they answer:

- **Sizing.** How large is the software project being estimated?
- **Technology.** How productive are the developers?
- **Effort and Schedule Calculation.** What amount of effort and time are required to complete the project?
- **Constrained Effort/Schedule Calculation.** How does the expected project outcome change when schedule and staffing constraints are applied?
- **Activity and Labor Allocation.** How should activities and labor be allocated into the estimate?
- **Cost Calculation.** Given expected effort, duration, and the labor allocation, how much will the project cost?
- **Defect Calculation.** Given product type, project duration, and other information, what is the expected, objective quality of the delivered software?
- **Maintenance Effort Calculation.** How much effort will be required to adequately maintain and upgrade a fielded software system?

Software Sizing

Software size is a key input to any estimating model, SEER-SEM being no exception. Supported sizing metrics include source lines of code (SLOC), function-based sizing (FBS) and a range of other measures. They are translated for internal use into effective size (S_e). S_e is a form of common currency within the model and enables new, reused, and even commercial off-the-shelf code to be mixed for an integrated analysis of the software development process. The generic calculation for S_e is:

$$S_e = \text{NewSize} + \text{ExistingSize} \times (0.4 \times \text{Redesign} + 0.25 \times \text{Reimpl} + 0.35 \times \text{Retest})$$

As indicated, S_e increases in direct proportion to the amount of new software being developed. S_e increases by a lesser amount as preexisting code is reused in a project. The extent of this increase is governed by the amount of rework (redesign, re-implementation, and retest) required to reuse the code.

Function-Based Sizing

While SLOC is an accepted way of measuring the absolute size of code from the developer's perspective, metrics such as function points capture software size functionally from the user's perspective. The function-based sizing (FBS) metric extends function points so that hidden parts of software such as complex algorithms can be sized more readily. FBS is translated directly into unadjusted function points (UFP).

In SEER-SEM, all size metrics are translated to S_e, including those entered using FBS. This is not a simple conversion, i.e., not a language-driven adjustment as is done with the much-derided *backfiring* method. Rather, the model incorporates factors, including phase at estimate, operating environment, application type, and application complexity. All these considerations significantly affect the mapping between functional size and S_e. After FBS is translated into function points, it is then converted into S_e as:

$$S_e = L_x \times (\text{AdjFactor} \times \text{UFP})^{(\text{Entropy}/1.2)}$$

where,

L_x is a language-dependent expansion factor.
AdjFactor is the outcome of calculations involving other factors mentioned above.
Entropy ranges from 1.04 to 1.2 depending on the type of software being developed.

Effort and Duration Calculations

A project's effort and duration are interrelated, as is reflected in their calculation within the model. Effort drives duration, notwithstanding productivity-related feedback between duration constraints and effort. The basic effort equation is:

$$K = D^{0.4}(S_e/C_{te})^{1.2}$$

where,

S_e is effective size – introduced earlier.
C_{te} is effective technology – a composite metric that captures factors relating to the efficiency or productivity with which development can be carried out. An extensive set of people, process, and product parameters feed into the effective technology rating. A higher rating means that development will be more productive.
D is staffing complexity – a rating of the project's inherent difficulty in terms of the rate at which staff are added to a project.

The general form of this equation should not be a surprise. In numerous empirical studies, the effort-size relationship has been seen to assume the general form $y = a \times \text{size}^b$ with a as the linear multiplier on size, and the exponent ranging between 0.9 and 1.2 depending on available data. Most experts feel that $b>1$ is a reasonable assumption, translated as *effort increases at a proportionally faster rate than size*. While SEER-SEM's value of 1.2 is at the high end of this range, the formula above is only part of the estimating process.

Once effort is obtained, duration is solved using the following equation:

$$t_d = D^{-0.2}(S_e/C_{te})^{0.4}$$

The duration equation is derived from key formulaic relationships (not detailed here). Its 0.4 exponent indicates that as a project's size increases, duration also increases, though less than proportionally. This size-duration relationship is also used in component-level scheduling algorithms with task overlaps computed to fall within total estimated project duration.

Time/Schedule Tradeoffs

In software projects, a limited exchange can be made between required effort and schedule. In fact, SEER-SEM optimizes according to minimum time or optimal effort scenarios. The first implies that a software project will staff aggressively to finish in the minimum amount of time, while the alternative permits schedule slippage for the sake of effort savings. The trade between minimum time and optimal effort is shown in Figure 1.

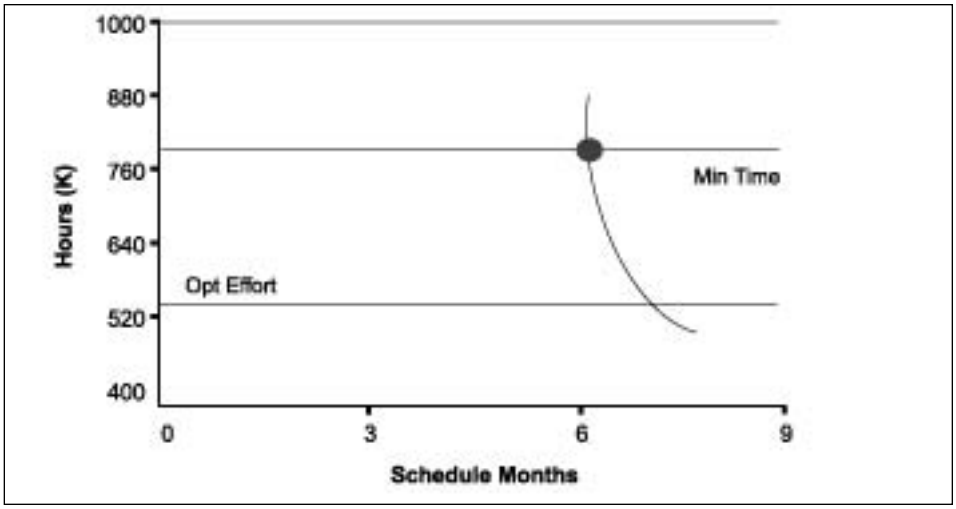


Figure 1: *Effort Schedule Tradeoff*

Staffing Constraints

Oftentimes specific staffing levels need to be factored into an estimate. Other factors aside, lower staffing leads to higher productivity per programmer while increased staffing reduces productivity. The dynamic relation between staffing and productivity can be described by an optimal staffing curve as shown in Figure 2.

The curve depicts optimal staffing over time for an idealized project. Its shape varies depending on project size and complexity. Areas around the curve illustrate the impact on individual productivity when staffing at any time varies from optimal. When staffing is too high, there is a productivity penalty as increased coordination is required while more staff must spend time getting up to speed. When staffing is too low, productivity increases due to tighter coordination among fewer staff and from team members who on average are more expert. Adding more staff may increase a team's ability to get work done but every additional person added is slightly less effective than the last.

Detailed Allocations of Effort and Duration

Project planners often need to know how a project's overall estimated effort and duration are allocated into specific activities and labor categories. While allocations are partially determined by patterns seen in past projects, they will vary for each project according to its unique characteristics. For example, there may be more or less requirements activity, testing, etc. Table 1 (see next page) provides a typical allocation, by percentage, of project effort into a matrix of labor types and activities.

Calibrating SEER-SEM

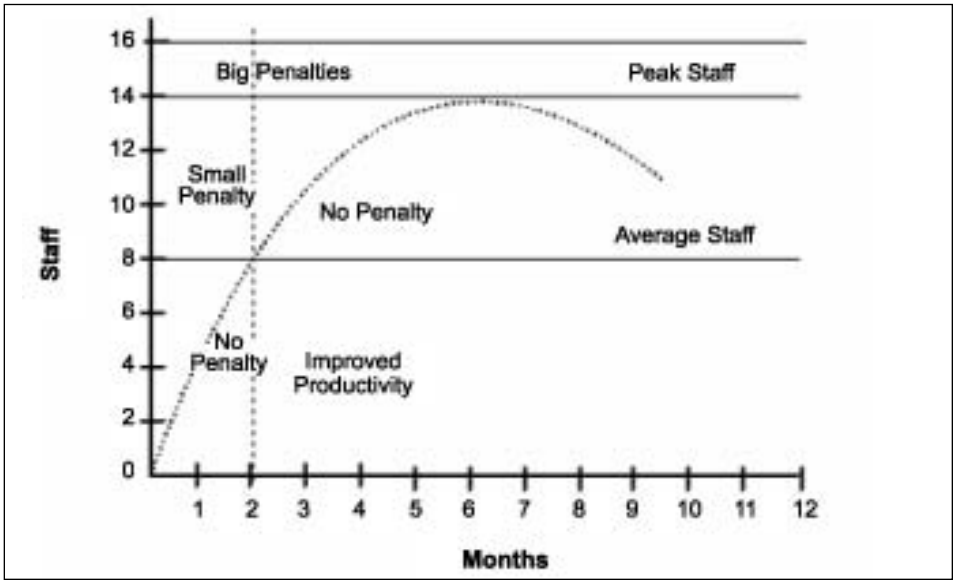
Key components of the SEER-SEM model have been described, but we have not discussed how it adapts to accurately estimate particular development scenarios, and how the model is kept current as software development technologies and methodologies evolve. The answer is simple: masses of ongoing research and analysis.

The modeling team regularly combs through raw data and industry studies to determine the latest trends and their impact on project productivity. As part of this effort, Galorath maintains a software project repository of approximately 6,000 projects (and growing). About 3,500 projects containing effort and duration outcomes are stored in a unified repository that can be readily accessed for studies. These are from both defense and commercial sources representing many development organizations, permitting calibration of the model to a wide array of potential projects. Additional project outcomes, in the hundreds, are also available to the company, which has also collected sizing and other information on thousands of additional projects.

Analysis involves running project data through SEER-SEM using a special calibration mode. The model is essentially run backwards to find calibration factors. These factors are evaluated across different data attributes (e.g. platform, application, etc.) to detect trends. A variety of methods are used to mitigate outlier data points and control for variation. The variance in the data set is also used to establish default parameter ranges; nearly all settings accommodate risk. Model settings are updated as new trends are established.

Galorath's work also is leveraged with findings from outside studies. For example, when examining relative language pro-

Figure 2: *Optimal Staffing Over the Project Life Cycle*



Labor Categories								
Activities	Management	Software Requirements	Design	Code	Data Prep	Test	Configuration Management	Quality Assurance
System Requirements Design	0.2%	0.7%	0.2%	0.0%	0.1%	0.2%	0.0%	0.0%
Software Requirements Analysis	0.5%	1.8%	0.6%	0.3%	0.3%	0.5%	0.1%	0.1%
Preliminary Design	0.9%	0.9%	3.6%	1.0%	0.7%	1.2%	0.2%	0.2%
Detailed Design	1.6%	1.5%	6.0%	1.8%	1.2%	2.1%	0.3%	0.3%
Code and Unit Test	1.6%	0.7%	1.4%	12.8%	1.4%	3.5%	0.9%	0.9%
Component Integrate and Test	2.2%	0.6%	1.1%	10.9%	2.2%	8.1%	1.4%	1.4%
Program Test	0.3%	0.1%	0.2%	1.3%	0.3%	0.9%	0.2%	0.2%
System Integrate Thru OT and E	1.4%	0.3%	0.7%	3.2%	0.2%	9.8%	0.9%	0.3%

Table 1: *Allocation of Activities and Labor for a Sample Project in SEER-SEM*

ductivity, the company first uses its repository to empirically determine the impact of using different languages. However, because not all languages are well covered, it turns to outside sources that provide language descriptions, evolution trees, multidimensional comparisons, etc. Putting all this information together permits the company to make informed judgments about even rarely occurring languages.

Cost estimation models must be able to estimate a wide array of projects. This is accomplished with a significant number of modeling instruments, most of which can be independently set by the user:

- **Sizing Measures.** Software's effective size varies according to many factors, and these factors change over time. As new languages are added to the developer's toolbox and old ones evolve, language mappings get updated. Sizing proxies also permit entirely new metrics to be added.
- **Knowledge Bases.** New platforms (or operating environments) and applications are regularly being identified and added to SEER-SEM by way of its knowledge bases. Knowledge bases actually represent collections of parameter settings. Parameters in turn cover many different facets of the development process and of a software product's potential characteristics; new platforms and applications usually can be defined with a collection of parameter settings.
- **Allocations.** According to project type, the balance shifts between types of activities and labor. Within SEER-SEM, detailed activity milestone and labor allocation tables are used to establish baseline allocations, which are then further adjusted depending on project-specific settings related to requirements, testing, and so forth.
- **Internal Calibrations.** Several internal instruments, both linear and nonlinear, permit high-level, systematic adjustments to estimates.

Beyond the Model

While this article has dealt exclusively with the core SEER-SEM model, other aspects of the tool are critically important to its practical application. Among its key design philosophies is the use of qualitative rating scales, user-selectable knowledge bases for basic calibration, and a work breakdown structure that differentiates between the system, program, and component levels. The SEER-SEM model will itself soon be complemented with a data mining system that produces entirely dynamic, data-driven estimates.◆

About the Authors

Lee Fischman is Special Projects director at Galorath Incorporated, where he develops new concepts, produces new software applications, and conducts research projects. His research interests include software metrics theory and novel applications of estimating algorithms. He received a Bachelor of Arts in economics from the University of Chicago and a Master of Arts in economics from University of California Los Angeles.

Galorath Incorporated
100 N Sepulveda BLVD
STE 1801
El Segundo, CA 90245
Phone: (310) 414-3222
E-mail: info@galorath.com

Karen McRitchie is vice president of Development at Galorath Incorporated, and is responsible for design, development and validation of current and new System Evaluation and Estimation of Resources tools. For her longstanding contribution to commercial cost prediction tools, the International Society of Parametric Analysts honored her with its 2002 Parametrician of the Year award. McRitchie has a Bachelor of Arts in mathematics and system science from the University of California Los Angeles, and completed Master of Art degree work at California State University, Northridge.

Galorath Incorporated
100 N Sepulveda BLVD
STE 1801
El Segundo, CA 90245
Phone: (310) 414-3222
E-mail: info@galorath.com

Daniel D. Galorath founded and is president of Galorath Incorporated. He has solved a variety of management, costing, systems, and software problems, performing all aspects of software development and management. His company has developed tools, methods, and training for software cost, schedule, risk analysis, and management decision support, including the industry standard System Evaluation and Estimation of Resources-Software Estimating Model. Galorath has a Bachelor of Arts and a Master of Business Administration from California State University, Dominguez Hills.

Galorath Incorporated
100 N Sepulveda BLVD
STE 1801
El Segundo, CA 90245
Phone: (310) 414-3222
E-mail: info@galorath.com